STUDIES IN SYSTEMATIC INSTRUCTION AND TRAINING

The Relationship between Algorithmic Processes for Instruction and Computer Models

Richard F. Schmid

Vernon S. Gerlach

Miroslav Valach

USAF Office of Scientific Research

Grant No. 76-2900

ARIZONA STATE UNIVERSITY
EDUCATIONAL TECHNOLOGY

Technical Report #81203

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(19) REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

(18) 1. REPORT NUMBER: AFOSR-TR-80-0230	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER: (9)	
(6) 4. TITLE (and Subtitle): The relationship between algorithmic processes for instruction and computer models.	5. TYPE OF REPORT & PERIOD COVERED: ~~Technical Report~~ Interim rept.
(14) TR-81203	6. PERFORMING ORG. REPORT NUMBER
(10) 7. AUTHOR(s): Richard F. Schmid, Vernon S. Gerlach, Miroslav Valach	(15) 8. CONTRACT OR GRANT NUMBER(s): AFOSR-76-2900
9. PERFORMING ORGANIZATION NAME AND ADDRESS: College of Education, Arizona State University, Tempe, Arizona	(16) 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F, 2313/A2, (17) A2
(11) 11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Office of Scientific Research/NL, Building 410, Bolling AFB, DC 20332	12. REPORT DATE: 3 Dec 1978
	13. NUMBER OF PAGES: 36
(12) 37 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer
Algorithm
Instruction
Training
Cognition

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Computer terminology and modeling are employed in an attempt to precisely describe human algorithmic behavior. Following the lead of many contemporary cognitive psychologists (e.g. Anderson & Bower; Quillian; Simon), the components and the structural relations of teaching and learning algorithms are cast into a systematic model. Within the model, the concepts of an algorithm's depth and width, and the components' processing order as parallel or serial are defined and examples are presented. The model not only maps the logical order

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED 409989

of events, but also allows the efficiency of various configurations to be calculated. Next the model's control activities are described, with special attention given to the level of synchronization of processing units or branches as well as the corresponding gate-keeping functions. Finally, the results of two experiments are compared with the model to ascertain its heuristic value. Most importantly, it was found that use and retention of algorithmic problem-solving strategies are best dealt with in a serial fashion. It is also hypothesized that the length of the serial string must remain short and that the algorithm must be dealt with in "chunks" rather than as a single unit. The dimension of the algorithm and the individual component's relations are easily described within the context of the model. A re-arrangement of the semantic content of the algorithm, a variable beyond the descriptive scope of the model, is also analyzed. Again in the domain of cognitive research, it was found that the logic or familiarity of the content of the discriminators and operators has a profound effect on learning but that neither logic nor familiarity produces differential effects along a concreteness dimension in the training stage. Finally, the beneficial nature of this modeling procedure for instructional designers is discussed.

Rule Learning and Systematic Instruction in Undergraduate Pilot Training

Vernon S. Gerlach, Principal Investigator

THE RELATIONSHIP BETWEEN ALGORITHMIC PROCESSES FOR INSTRUCTION AND COMPUTER MODELS

Richard F. Schmid
Vernon S. Gerlach
Miroslav Valach

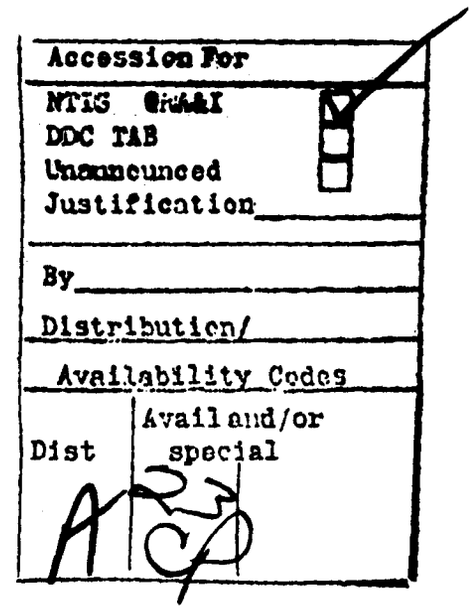
Technical Report #8120

Research sponsored by the Air Force Office of Scientific Research, Air Force Systems Command, USAF, under Grant No. AFOSR 75-2800. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

College of Education
Arizona State University
Tempe, Arizona

December, 1978

THE RELATIONSHIP BETWEEN ALGORITHMIC PROCESSES FOR INSTRUCTION AND COMPUTER MODELS



The Relationship between Algorithmic Processes and Computer Models

"I think, therefore I am." -- Descartes (1637)

The use of computer logic as an explanatory vehicle for psychological processes is a widely used and productive practice (Anderson & Bower, 1973; Feigenbaum, 1963; Newell & Simon, 1961; Quillian, 1968). The development of algorithms in instructional theory has also benefited both directly and indirectly from computer science. Computer-type programs, which are by definition "pure" algorithms, have been directly integrated into the classroom as teaching devices (Landa, 1974; Scandura, Durnin, Ehrenpreis, & Lugar, 1971; Schmid & Gerlach, 1977; Schmid, Portnoy, & Burns, 1976). The indirect use of computer logic via the science of cybernetics has assisted theoreticians in their formulations of algorithmic learning theories (Bunge, 1968, 1971; Landa, 1974, 1976; Lansky, 1969). A logical extension of this scientific evolution is the use of computer algorithms and computer language to assist in explaining human behavior which can be classified as algorithmic. The following report begins with an examination of a computer model of processes which are algorithmically defined. Then empirical data generated by two studies conducted in our laboratory are examined in the context of the model to test whether or not its content and structure accurately simulate human behavior.

Human Behavior and Computer Behavior

Humans have the ability to generate and transform ideas into self-serving tools. The computer, on the other hand, is just a tool, and is conceptually no more significant than its developmental predecessors (groups of pebbles or sticks, abacus, or slide rule). The mystique which

has developed around computers is a product of their complexity and their ability to "think" far faster and more accurately, thus "better," than we. Indeed, it is the depth and versatility of computer processes that have led researchers to infer similarities between computer thought and human thought. However, before employing computer models as a means of describing and perhaps understanding human cognition, two restricting principles must be recognized.

First, computers were developed to improve upon our own process of thinking and were modeled to support and free human thought. Thus, the present day computer is completely dependent upon preprogramed logic and can behave only in certain restricted ways. For example, if the computer were asked to drive an automobile from point A to point B and the road were blocked, it would not know what to do unless human thought (i.e., a thinking programmer) had introduced the concept of a detour into the computer's memory and supplied it with all the tools necessary for completing the task: inspecting an appropriate map, choosing the optimal detour, turning around, and knowing when reintersection with the original route or destination has occurred. The computer's ability to imitate human behavior may be profound only to the extent to which it is both understood and permitted to do so by us, its programmers. To assume that computers can wholly reproduce our mental activities given state-of-the-art technology may be naive; such an assumption could encumber or inhibit the examination of alternative descriptive systems.

The second restriction is our own limitation in simulating cognition. To date (this report not excepted), man has approached the use of computers in much the same way he did the problem of flying. We observe and analyze present modes of thinking, just as scientists observed the bird, and try

to imitate the critical factors, hoping to develop a replica of the phenomenon. It may be that, as in the study of flying, we are trying to understand the process by simply observing and imitating rather than searching at the same time for an "unobservable" variable which explains how human thought occurs. Like our courageous forerunners in aviation, the authors of this report are still observing the bird while collecting and evaluating data for later "unobservable" discoveries. However, we have no intention of leaping off a precipice in a strong wind. The comparison created between computer programming characteristics and our own thought processes will serve as an analog of the two algorithmic behaviors. The analog, we hope, will assist instructional designers in predicting and planning how learners can best acquire and retain new information.

The Model

Four basic concepts must be defined and illustrated to introduce the model. When developing a computer program, these concepts interact in an orderly fashion to determine the effectiveness and efficiency of the program's operation. After analyzing the model's structure, we will apply its terminology and logic to the known characteristics of algorithmic instruction. Finally, we will compare the operating procedures of the model with the results of two experiments on algorithmic instruction.

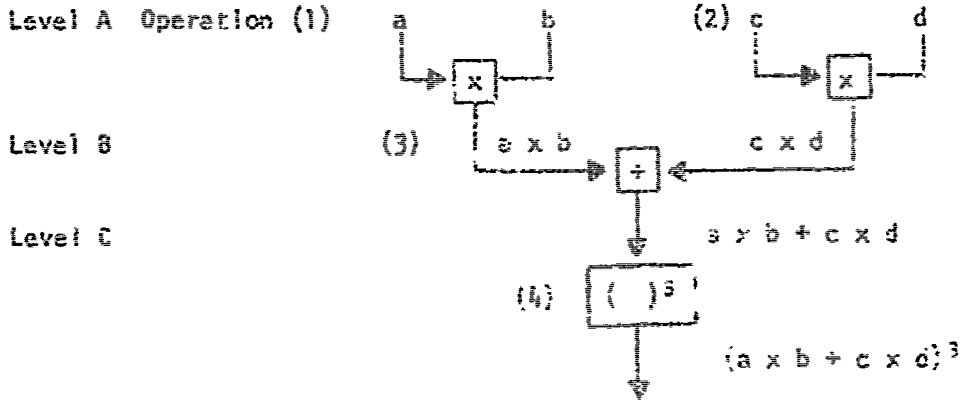
Structure of the model. Of the four concepts dealt with in this section, the terms depth and width refer to physical characteristics of our program, while the terms serial and parallel refer to processing characteristics. The physical concepts can be perceived when presented

in the form of a conventional flowchart. The processing characteristics, however, require an understanding of the problem being solved and the nature of the medium (a human, one or more connected computers, etc.) through which the program is run. Let us consider several types of mathematical problems in order to explain these characteristics.

If we wish to compute an answer for the equation

$$y = (ab + cd)^3 \qquad (1)$$

and if (1) is interpreted algorithmically (.e., the algorithm serves as a prescription for the process), it is immediately apparent that processors for adding, multiplying, and raising to the third power must be available. In the information flow diagram (2) this could be expressed



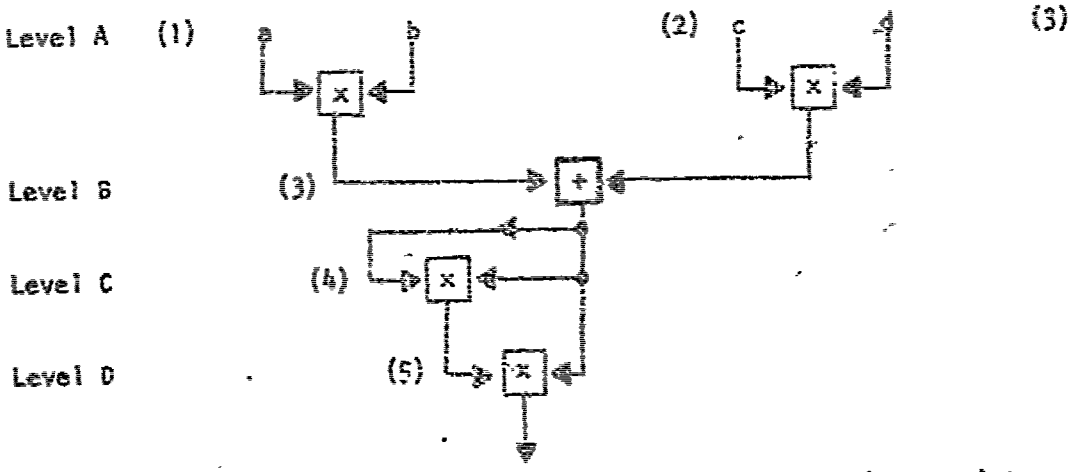
Since the algorithm has three levels, A, B, and C, its depth is three. The depth is, simply, always equal to the number of levels. The width, in this example, is two. The width of Level A is two, of Level B one, and of Level C one. The width of an algorithm is always equal to that of the "widest" level.

The width of the algorithm is associated with the notion of how many parallel processes can be recognized at the particular level. The depth defines the serial processing required for its execution. Since mutually

independent variables (i.e., input variables) are used in Operations 1 and 2, the two processes (Operations 1 and 2) are independent. We call them parallel because it does not matter whether or not they are computed simultaneously; nor, if they are not computed simultaneously, does it matter which of the two is computed first.

In contrast, Operation 3 is serial to the set of Operations 1 and 2, since it can be completed only after Operations 1 and 2 are completed.

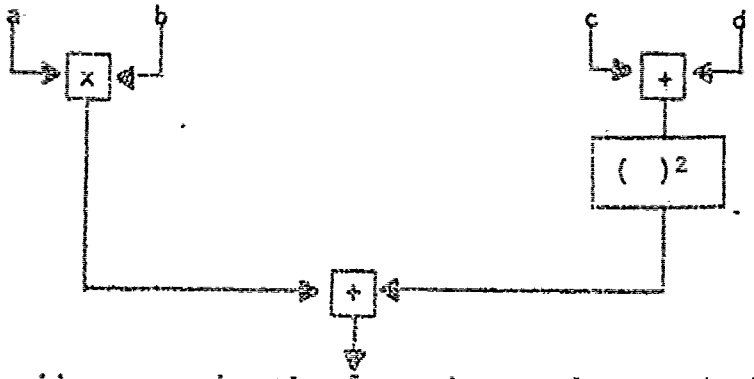
Suppose, however, that we want to compute $y = (ab + cd)^3$ in a situation where only adders and multipliers are available; we do not have a processor capable of raising a quantity to the third power. In diagram form (3), the process is expressed as



In this case, the width of our operation will remain two (1 and 2) but the depth will increase to four (Levels A, B, C, and D). Operations 1 and 2 can be processed in parallel, while the operations in Levels B, C, and D must be processed serially. In each diagram, the overall algorithm is both serial and parallel.

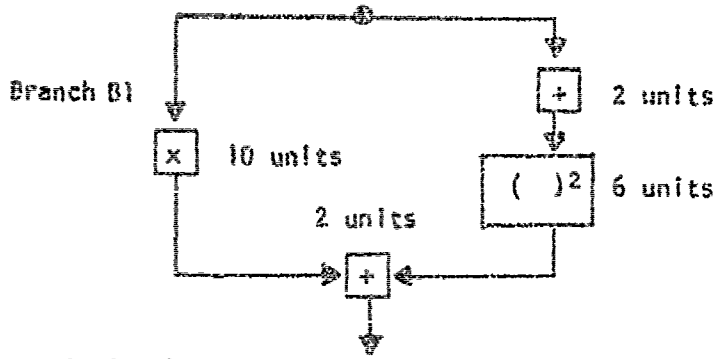
Processing efficiency. Next let us consider how to design the process which requires the least time. If we wish to optimize the process, we must use processing time as our criterion. Consider the following statement and its corresponding diagram:

$$y = ab \div (c + d)^2 \tag{4}$$



(5)

When we consider processing time for each operation, we obtain, for example, the following:



(6)

From the diagram it is obvious that the processing time is the "race" between branches B1 and B2, and that the larger of the two yields the total time (here B1, which requires 10 time units).

It is also obvious that if the power-two processor [( )^2], with six units of processing time, is replaced by a multiplier with 10 units of processing time, as in B1, then B2 would determine the total processing time; in the above illustration, four additional units would be added to B2 (a total of 14 units). Thus considered, the optimal distribution of

mutually independent tasks would be to "balance" the branches, or to employ more efficient processors where applicable. In any case, the total processing time can be calculated (see Mantel, Note 1 for a similar treatment of efficiency modeling).

Another illustration of optimizing is provided by the following example. Assuming that one adder can add only two numbers simultaneously, the minimum number of processors and the minimum processing time required to compute

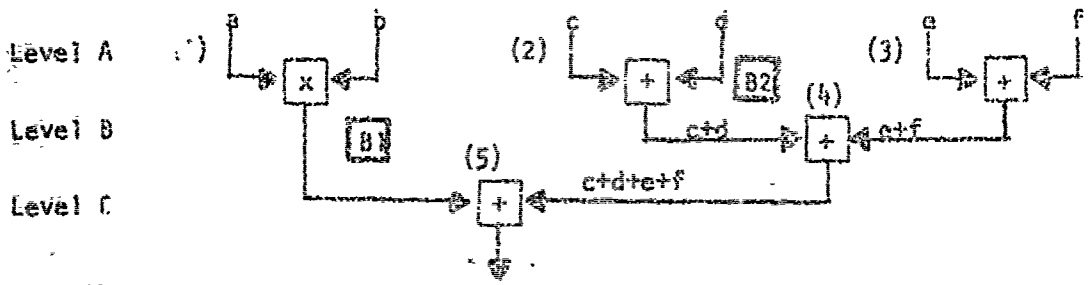
$$y = ab + c + d + e + f \tag{7}$$

is contingent upon the difference between the processing times of the multiplier and the adder. Consider the following cases:

Multiplication time (T_m) = Addition time (T_a) = 2 units of processing time.

In diagram form (8), this is expressed

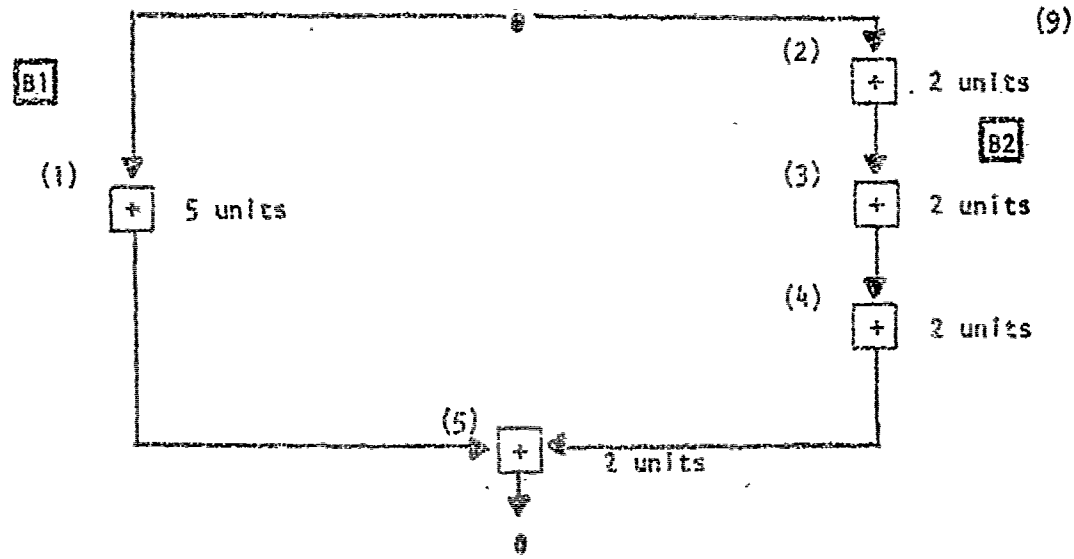
(8)



Since the depth is three, and since each of the three levels requires two units of processing time, the total processing time is six units. The procedure illustrated in the diagram above is optimal as measured by the processing time and by number of processors; adding any additional processors would not shorten the total time and using less processors would not shorten it either. When used for minimal hardware design, the diagram shows that at Level A there is need for one multiplier and two adders. At

Level B there is only one adder; one of the two from Level A can be used again. The issue is true of Level C, where one of the two adders at Level A can be used. Thus, the total minimal hardware required to yield the shortest possible total computation time will consist of one multiplier and two adders.

Consider the diagram (9), in which $T_m = 5$ and $T_a = 2$, where Branch B2 is designed for utilizing the same adder three successive times.



It is evident that the computation in Branch B2 will require one more unit than will B1. Therefore, using one adder three times in B2 is not the shortest possible arrangement. To keep the total time to a minimum, both parallel adders of the first level of Branch 2 [as in (8)] are still justified. On the other hand, the diagram (8) is justified on the basis of the number of processors: one multiplier and two adders, one of which is used once, the other three times.

Finally, consider a case in which $T_m = 10$ and $T_a = 2$. In this case, one adder is sufficient in Branch 2, leaving four unused units of processing time. The total time is 12 units; two processors, one multiplier, and one adder are required for minimal total computational time.

Parallel versus serial processing. The computer is capable of working with operations within the same task in two discrete ways.

(1) Parallel processing is the type which occurs during the processing of a given task. Two constraints govern parallel processing. Parallel processing is appropriate only when

(a) there is more than one processor available (this is a design constraint) and when

(b) the approach to solving the problem permits parallel operations (this is a method constraint).

The first constraint can be illustrated by means of a negative example: If I have only one adder, I cannot add by using three adders in parallel. The second constraint can be illustrated by an example involving computation of a trajectory. If the approach selected involves numerical computation of differential equations, I must use serial, not parallel, processing; the seriality, however, lies not in the problem per se but rather in the method I have selected for determining the trajectory (i.e., the method of numerically computing differential equations).

(2) Serial processing, as its name suggests, involves a sequential approach to problem-solving. Because parallel processing can be both more efficient and perhaps more powerful, it is also subject to a greater variety of constraints than is serial processing. As a consequence, if one cannot process a task in parallel, one still might be able to do it

serially. In contrast, if one cannot process a task serially, it cannot be completed until one employs different approach to the solution. This is, of course, true only of digital processing as opposed to analog processing.

In strictly serial processing, where the serlality is established by the method used, it is not possible to rearrange the order of subsequent operations. Total time or cost or effectiveness as measured by a given parameter can rarely be optimized since there is, generally, insufficient freedom for alteration or manipulation of the elements in the process. Each step is dependent upon the successful completion of the preceding step.

Depth and width in parallel processing. Two concepts discussed above are central to the discussion of parallel processing, and are particularly relevant to the remainder of this report. The first is depth: how long an algorithm is needed, or how many operations are necessary to complete a process? The concept of depth is related to processing time. The second is width: how many branches (more specifically, how many parallel branches) can be processed simultaneously or independent of one another at at least one level? This concept is related to the maximum number of processors operating simultaneously.

In computer programming, we are generally concerned with transcribing an algorithm which is quasi parallel into a truly serial algorithm. The result is the "program." However, even though the computer may (and in most cases does) perform the process serially, the programmer initially benefits from conceiving and writing the program in parallel form. The language used and the hardware capabilities then determine the width and depth of the program. More powerful languages enable the programmer to utilize "parallel process" configurations to a far greater extent than do weaker ones.

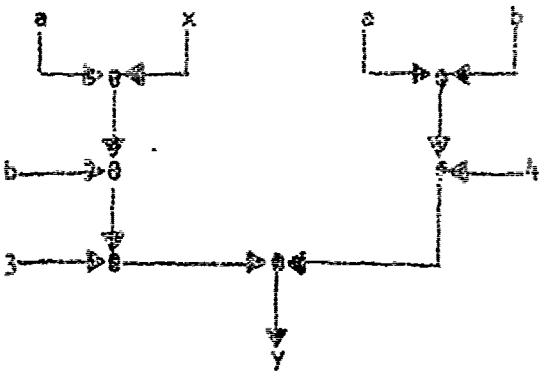
The merger of the two processes, serial and parallel, can be labeled serioparallel. When we consider arithmetic or logic expressions, serioparallel processing is properly restricted to processes which can be described as expressions in the language of arithmetic or logic. Some natural language sentences manifest serioparallel structure in this sense. The strings of operations, variables, and parentheses must be subject to the constraints of the language. In addition, the size or the number and range of computer processors and their interrelations limit the extent to which the language can be applied. Advances in both areas are such that any detailed description here would soon be outdated. Presently, large computers (with more than one CPU) permit parallel processing of two or more independent tasks. However, efforts have also been made to increase effectiveness through parallel processing of different sections of one program.

An example of serioparallel processing is illustrated in the solution of the statement

$$y = (ax + b)^3 - 4\,ab \tag{10}$$

in diagram form

(11)



Consider another expression

$$y = [\ (ax + b)^3 - 4\,ab + \frac{ax + b}{5} + ax\ ]^2. \tag{12}$$

If we choose to use a non-redundant process for computing y, the following statement would apply.

$$q = ax + b \tag{13}$$

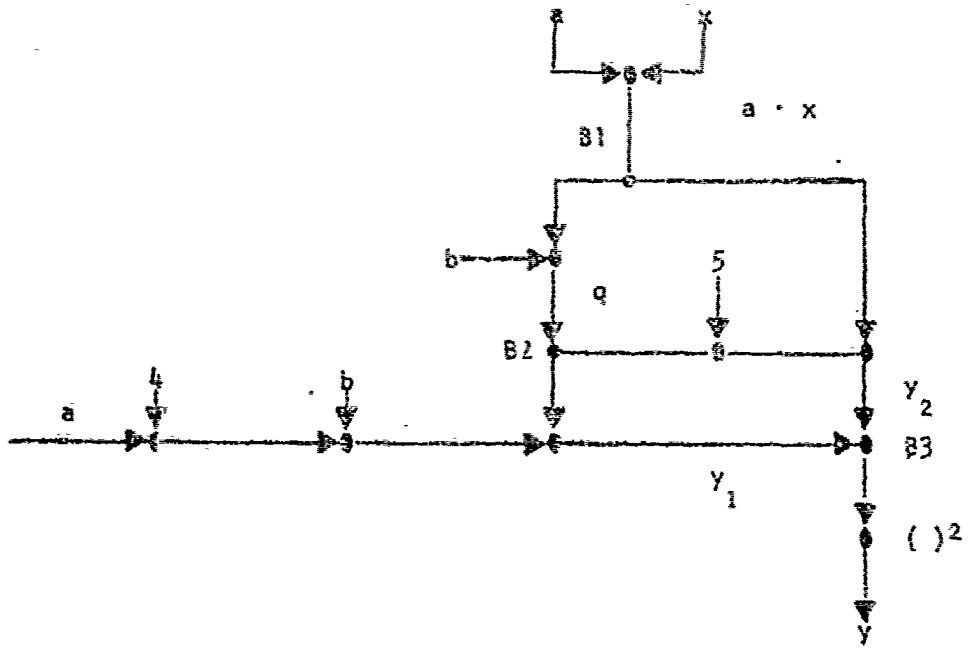
$$y_1 = q^3 - 4ab$$

$$y_2 = q/5 + ax$$

$$y = (y_1 + y_2)^2$$

Diagram (14), corresponding to statement (13), shows a process which is <u>not</u> serioparallel in terms of the arithmetic language.

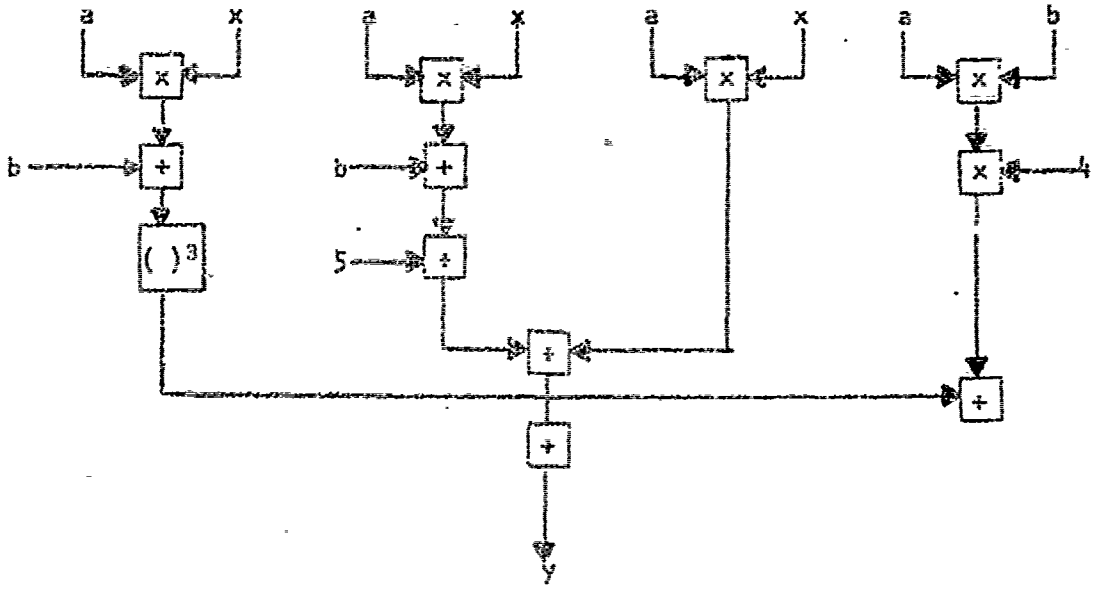
(14)



It cannot be serioparallel because of the two branchings (labeled B1 and B2), following which the variables (a · x) and q are processed in two different branches; and because the variables y_1, y_2 resulting, respectively, from the processes do indeed input the same process again following B3. Processes of this type cannot be rewritten into a single algebraic expression unless redundancies are introduced [see formula (13)]. Variable q is used as an input in both branches of variables y_1 and y_2. However, algebraic language, when interpreted as procedural language, does not permit the writing of an expression which branches in the middle and then returns for further processing (unless it returns to the same branch). The choice, then, is either one of using two or more expressions for describing the entire process as was done in (13), above, or using a single expression which is procedurally redundant, as in formula (12).

The diagram for formula (12) in fully redundant form is given for comparison in (15).

(15)



Control of the Activity in Digital Systems

In so called digital systems (as opposed to analog systems) internal activity moves forward on a step-by-step basis by transitions from one state to another. States of the system are significant when a description of the system activity is considered in which only stable states are subjects of the description while unstable states are ignored. Unstable states, also called transitions, are important considerations from a lower level (more refined) point of view.

Coordination of activities in a digital system can be accomplished by two methods, each of which is related to a class of systems: synchronized and asynchronized. Combinations of the two can be effected; however, they are not of concern here.

Synchronized systems. A synchronized system is one in which transitions from one state to another are made for the whole system (or for the synchronized

portion of the system) simultaneously. This is achieved by using a so-called central clock which is set for a certain frequency. A single period of the frequency is called a unit or a step of the clock. The length of the step is such that there is a sufficient amount of time for all transitions to move safely from the current stable state into the next stable state. The clock impulse or a signal derived from the clock is the triggering signal for starting the transitions.

Asynchronized systems. An asynchronized system is one in which processors or groups of elements make transitions independently of each other; each progresses at its own rate. As long as each group follows its own task, independent of all others, there is no reason to decrease or increase the pace. However, when signals resulting from different activities are to be processed together, it becomes necessary to coordinate the actions. Signals coming earlier have to wait for signals coming from the groups that needed more time to produce them. Internal control is that part of the system that provides the proper coordination of internal activities.

Separation of the control structure in an asynchronized system of processors. When more than one processor is present in a system, there is a need for coordination of the actions. This need gives rise to a control structure consisting of local units performing simple logical operations. The set of all such units is called a control structure.

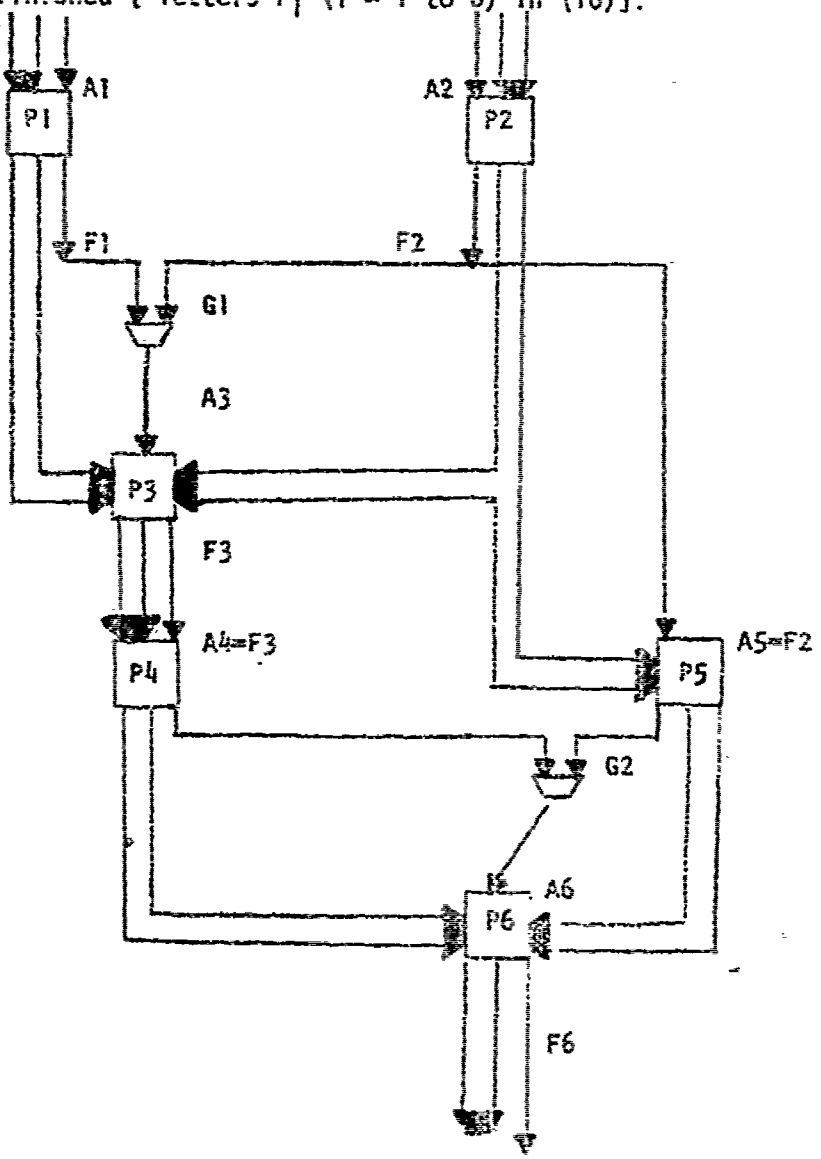
Control structure can thus be described as a logical structure that coordinates the activities of the processors engaged in their own mutually independent actions. In the simplest case, units of control structure do

not need to communicate with each other since a simple coordination does not require it. Control structure consisting of elements that work independently of each other is the lowest level of the hierarchy of control complexity.

Two kinds of signals in each processor of the system are of concern to this control activity:

(1) The signals that trigger the processor for initiating its activity [letters A_i (i = 1 to 6) in (16)] and

(2) The signals conveying the information that the activity of the processor is finished [letters F_i (i = 1 to 6) in (16)].



(16)

Example of an Asynchronized System and Its Control Structure

Diagram (16) shows a system of six processors, P1 through P6. Double lines represent the flow of data from one processor to another. There are two input data channels feeding the system from the outside (at the upper part of the figure) and one output data channel (at the lower part).

Each processor also has signals, shown as single lines: A-signal, which activates the processor, and F-signal, which is generated by the processor when the activity is finished (i.e., the output data is ready, waiting at the processor's output).

G1 and G2 are logical AND elements (gates). Signal A3 is generated only when both F1 and F2 are present at the input of the gate. Thus, the activity of the processor P3 will begin only after both processors P1 and P2 have finished their activity and have generated F1 and F2, respectively. Similarly, processor P6 will begin only after both P4 and P5 finish their activity, since G2 is an AND gate generating A6 only when both F4 and F5 appear at the input.

Symbolic Representation of the Control

Relations between A and F signals of the above system S can be symbolically described using Boolean algebra. Then

$$A3 = F1 \wedge F2 \tag{17}$$

$$A4 = F3$$

$$A5 = F2$$

$$A6 = F4 \wedge F5$$

where all variables have two values (0, 1) with the following interpretation:

0 = the signal is not present and

1 = the signal is present.

More general description will result when gates G are introduced as Boolean functions. Then

$$A3 = G1\ (F1, F2) \qquad (18)$$

$$A4 = F3$$

$$A5 = F2$$

$$A6 = G2\ (F4, F5)$$

where gates are represented as Boolean functions, signals F are represented as their input variables (arguments), and signals A are represented as output variables. Thus, both equations (17) and (18) can be considered as symbolic representation of the control in S.

Application of the Model to Algorithmic Classroom Behavior

An underlying assumption of this report is that the same principles of rule application which govern the use of computers may also apply to certain human behaviors. The rules of interest are algorithmic in nature and can be seen to occur frequently in ordinary classroom procedures. The tasks with which these rules can be integrated range from simple discrimination (e.g., associating sounds with letters) to complex problem solving (e.g., medical diagnosis and prognosis), with the primary differences residing in the entry skills of the learner. Once the learner possesses the content and procedural base underlying the algorithm, all that is left to do is to master the sequence of events and their interrelations (Gerlach, Reiser, & Brecke, 1975). We will use the model to describe how the teacher can generate either learning or teaching algorithms with greater efficiency and how the result can increase learning effectiveness.

The research algorithm. The algorithm used in the studies we will be examining consisted of five discriminators and five different operators (see Figure 1). The algorithm is used to calculate the amount of tax owed or credited on stock transactions involving gains and losses. In terms of the model, the width of this algorithm is five: that is, there are five operators or discriminators on the same level. As stated earlier, the processing width of each branch always remains one for human operation. The depth of this algorithm is four, because the longest branch consists of four serial transitions, with individual branches ranging from three to four. Unlike the algorithms in the previous section, the tax problem entails five possible outcomes rather than just one. Nevertheless, the concepts and principles of the model apply with equal facility, as they will with any pure algorithm (as opposed to a quasi-algorithm, Landa, 1974; Bunge & Carrasco, 1977).

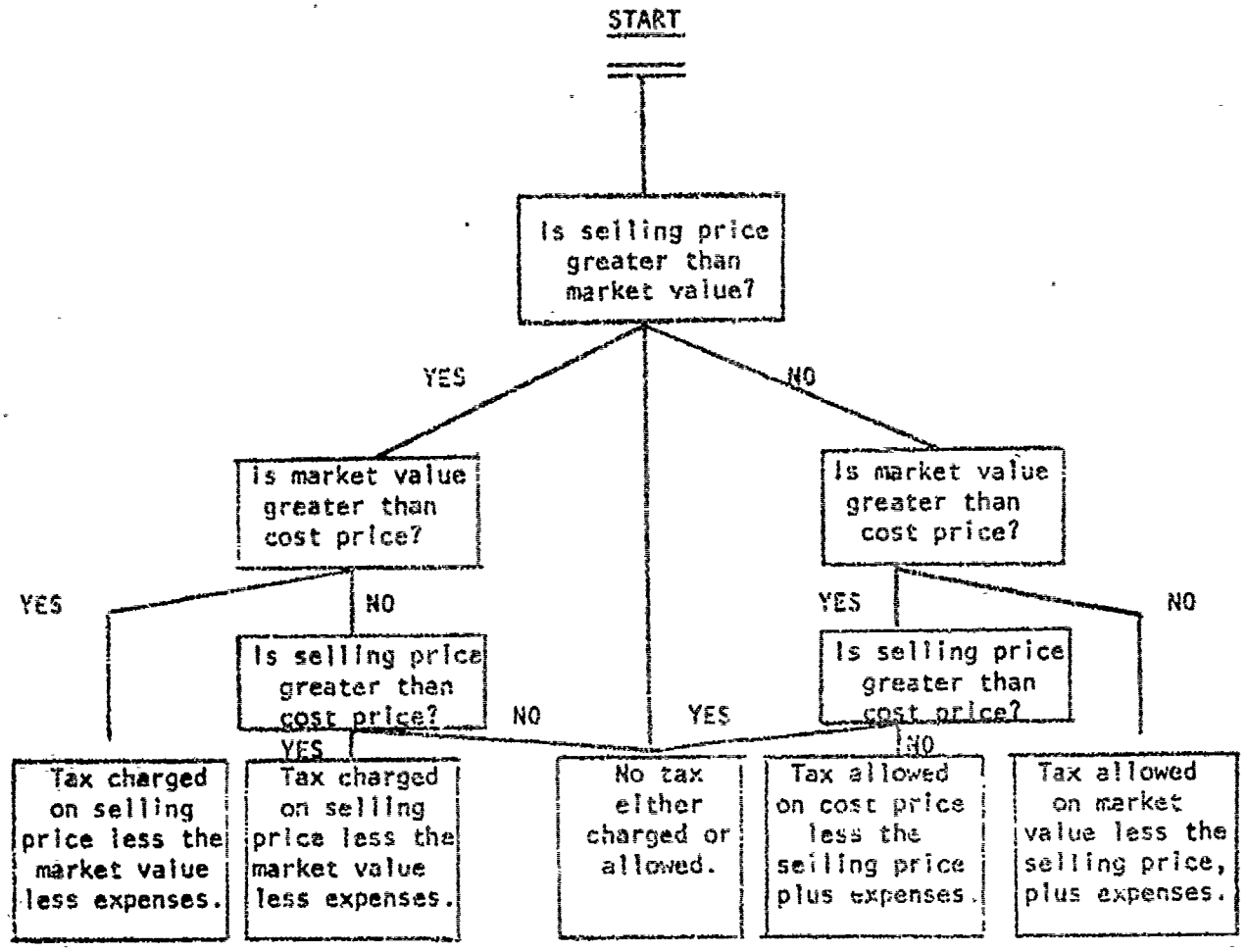


Figure 1.

Algorithm for solving tax computation problems (Schmid & Gerlach, 1977).

Next let us determine the processing characteristics of the tax problem. The processing descriptors cannot be calculated without an examination of the problem itself. To illustrate, it appears upon inspection of the flowchart that parallel processes can occur on Levels 2, 3, and 4. However, the following sample problem will demonstrate otherwise. Let us assume that stock XYZ was bought for $1000 and sold for $2000. The expenses amounted to $150. To calculate taxes, a fourth variable must be considered: the market value of the stock on April 15th, which is the theoretical worth of the stock. The market value is used in the following way. If you bought the stock at less than its theoretical value, the government gives you a break: you subtract the market value instead of the cost price to calculate profits. By the same token, if you buy a stock and its market value goes down and if you sell at less than the market value, the government allows you to deduct only the theoretical value because you made the mistake of buying an overpriced stock. Returning then to the sample problem, the market value was set at $750. Using Figure 1, the answer to (1) is Yes, which leads to (2) which is answered No, which leads to (4) which is answered Yes, which leads to (7). The called-for operations will yield the answer, "Tax charged on $850."

It is immediately evident that whenever a discriminator is employed, only one branch can be chosen. Algorithmic discriminators should always result in a single continuation which automatically excludes the remaining branches.[1] The tax problem as presented above is serial when passing

---

[1]It is, indeed, well known that any discriminator can be reduced to a set of binary discriminators. For example, a problem involving a traffic signal with a green, a red, and a yellow light is handled thus: Is the light green? (Yes-No) If No, is the light red? (Yes-No) If the answer to this question is No, the light must, obviously be yellow.

through the discriminators. The discriminators rely on the serial characteristic of dependence upon input from some prior information or signal producing component of the algorithm. In general, parallel processing can occur between the discriminators, or when the initial level of the algorithm includes more than one branch (as was always the case in the algorithms used to describe the model), or in the last (output) branches.

The final point to be made concerns the comparison of processing characteristics of humans and computers. The computer can be designed with an inherent processing-in-parallel capability, i.e., with more than one processor, and with the executive (control structure) for coordinating the system (for example 64 processors in ILLIAC IV). For all practical purposes, the human mind can concentrate on one task at a time. However, humans resemble computers inasmuch as they can operate on a time-sharing basis, where the solutions to intermittent transitions are derived and held in or drawn from memory for later use. The extent to which instruction can enable people to acquire the ability to use memory in this manner, as well as the relative efficiency and effectiveness of such training or instruction, have yet to be determined. While it is beyond the scope of the present report to examine the specifics of human parallel processing, it is our guess that it is this aspect of the model which will assist us in deriving a clearer understanding of the process of inference. We also suspect that parallel processing comes into play in quasi-algorithmic and heuristic activities. The first study discussed in this report only touches the surface of this problem.

Human thought and parallel processing. Much work has been done in the area of serial and parallel processing (Pask, 1976; Palvio, 1971),

but it has generally assessed individual differences and applied them to the aptitude-treatment interaction paradigm, which is demonstrably atheoretical (Snow, 1976). Our interest in parallel processing has been narrowed to whether algorithms are better taught in serial subcomponents, in parallel clusters, or a combination of the two. Schmid and Gerlach (1977) tested three teaching methods for the tax problem. Treatment P (prose) was a prose version of the algorithm; subjects were allowed to develop their own solution strategy following an essentially serial prompting. The subjects in Treatment FS (flowchart serial) were given an intact flowchart (Figure 1) and directed through the practice problems in an essentially serial fashion, solving the problems by using one complete branch at a time. Subjects in Treatment FU (flowchart unit) received the same flowchart as those in Treatment FS but were forced to memorize its content in a top-down fashion; information was withdrawn or faded by deleting the upper-levels first. The flowchart was therefore treated as a single solution unit.

The results of this study were quite dramatic. Students in Treatments P and FS performed equally well at a high level of mastery following only 12-15 minutes of instruction (approximately 85% achievement). However, the FU group, which was asked to learn the algorithm as a single unit, performed significantly less well on both immediate and one-week delayed tests (approximately 60%). It was assumed that, due to the similar achievement of the P and FS groups, those subjects were working with the same algorithm and adopted the same strategy. To accept this assumption, however, one would expect the FS subjects to work more efficiently because the instruction directed them immediately to the superior strategy. The

time data confirmed this hypothesis. The FS group mastered the algorithm in significantly less time than either of the other groups. The Prose group, once it adopted and learned the proper strategy, worked as efficiently as the Flowchart Serial group on the posttests; both groups worked faster than the FU group. Finally, one might expect the P subjects to have learned and retained the strategy better because their treatment forced them to formulate and utilize the serial strategy without formal prompting. The significant Representation (the three instructional treatments) x Availability (the algorithm was withheld or made available during the posttests) interaction on time data confirmed even this prediction. "...the flowchart group spent the same amount of time either with or without the procedure; the P group spent significantly less time without, and the FU subjects spent significantly more time without the procedure present. Further, the FS subjects worked significantly faster than the P subjects when the procedure was available, whereas the opposite held true when the procedure was removed" (Schmid & Gerlach, 1977, p. 20). Figure 2 ~~of that~~ Interaction, is included to illustrate the effect. Thus, it appears from these data that "forced processing" on a serial segmentation approach produces the best instructional combination.

Though this demonstration is greatly limited in scope, and thus generalizability, two further points can now tentatively be made using the width and depth concepts. If learning is in fact facilitated by a serial approach (and the vast quantity of mnemonic research suggests that it should), then teachers would be advised to construct algorithms of a restricted width when possible. Miller's (1956) classic memory load principle of 7 ± 2 might well be applied until directed research answers

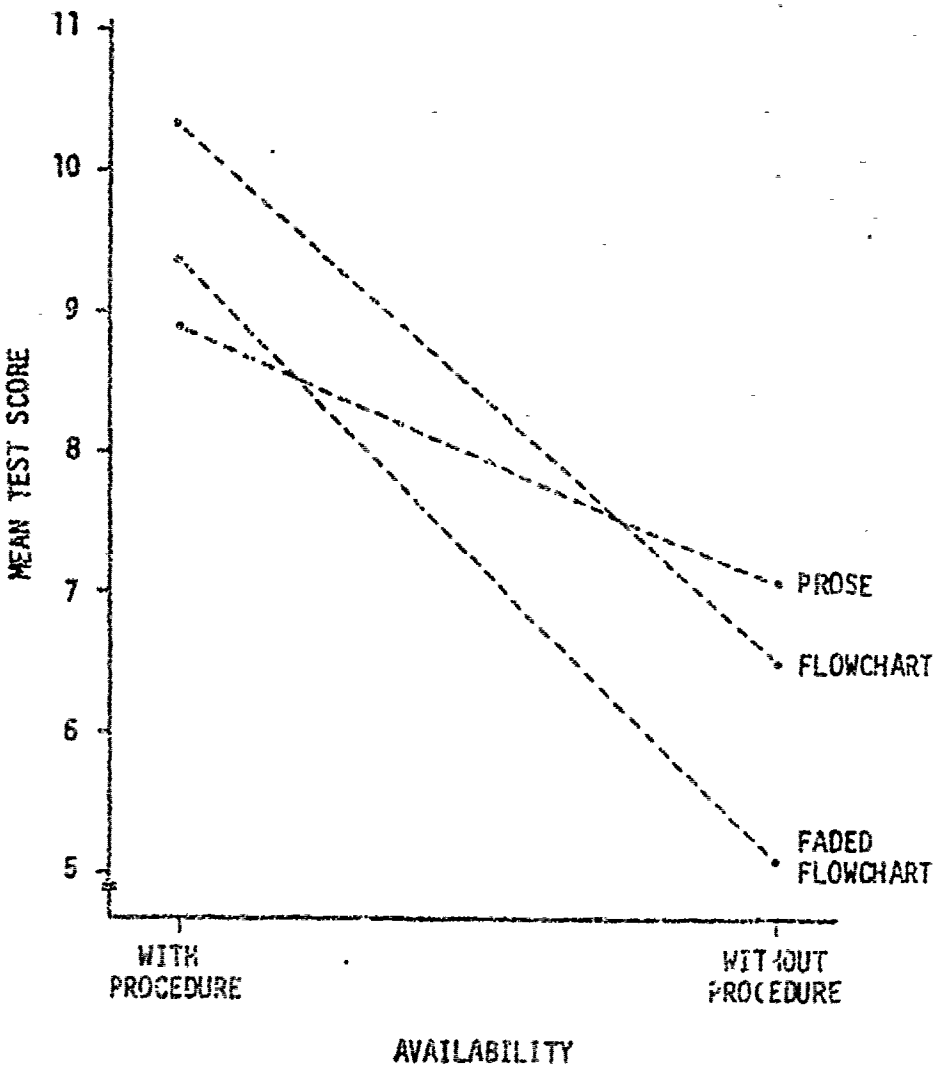


Figure 2. Representation x availability interaction from the repeated measures anova on test scores regrouped on delayed test availability

this problem. Furthermore, parallel transition points are not likely to be similar enough for effective chunking. The same principle can be implemented in developing serial (depth) strings. A pilot study also reported in Schmid & Gerlach (1977) demonstrated that an extremely lengthy serial array becomes unwieldy for the learner. Indeed, the above research only verifies Landa's (Note 2) basic instructional principle of breaking the algorithm down and teaching it in logical parts until mastered.

Beyond the mechanics. Thus far, we have described quantifiable variables which might affect learning, and provided some guidelines for using these factors to the learner's advantage. The superiority of serial tracking suggested to us that perhaps a qualitative factor may underlie, or at least influence, the serial/parallel difference. We also considered the possibility that this more basic factor, if controlled for, might equalize the obtained differences.

Serial processing entails a string of transitions, each serving as a link in a single chain, each drawing from and contributing to the appropriate adjacent transitions. Serial processes are therefore always intrinsically logical and orderly. However, the logical order of things is lost to the computer. Its only equivalent to entry skills is the amount of time a given processor requires to complete a given operation. Practically speaking, the computer either knows or it doesn't know, which means we, as users, either receive the answer very quickly with incredible accuracy, or not at all. Human thought usually falls somewhere between those extremes. To examine these shades of difference, in the second study (Schmid & Gerlach, 1978) we introduced human logic into an otherwise completely controlled algorithmic procedure. For example, the original

algorithm first asked, "Is the selling price greater than the market value?" A second algorithm solving exactly the same tax problem began with the question, "Did you make or lose money...?" (See Figure 3). The assumption was that the second question is more logical or familiar to most learners, thus easier to remember. This "logical" algorithm was written to possess the identical structural characteristics of the original: width = 5; depth = 3; requiring entirely serial processing, employing seven discriminators and five operators. The logical algorithm was a graphic mirror image, changing only the "human logic" by means of content alterations. To further test the intrinsic value of the logical reformation, both the original and logical treatments were taught using only letter symbols (e.g., "Is S > M?" for "Is the selling price greater than the market value?" and so on). The Sequence (original vs. logical) main effect reached marginal significance for combined immediate and delayed posttest scores, and significance at the .02 level on the immediate test alone, both in favor of the logical algorithm. There were no confounding interactions with the Form (verbal vs. symbolic) factor. The time data provided even stronger differences, with both combined and separate posttest times in favor of the logical group.

These data, while experimentally very satisfying, were not unexpected. The more familiar a learner is with the material, the better the material will be learned (Schmid, Note 3). The absence of an interaction between the presentation formats further suggests that the effect of familiarity is general rather than specific to any particular algorithmic representation. From the standpoint of the model, content familiarity appears to improve the effectiveness not only of the discrete processing units

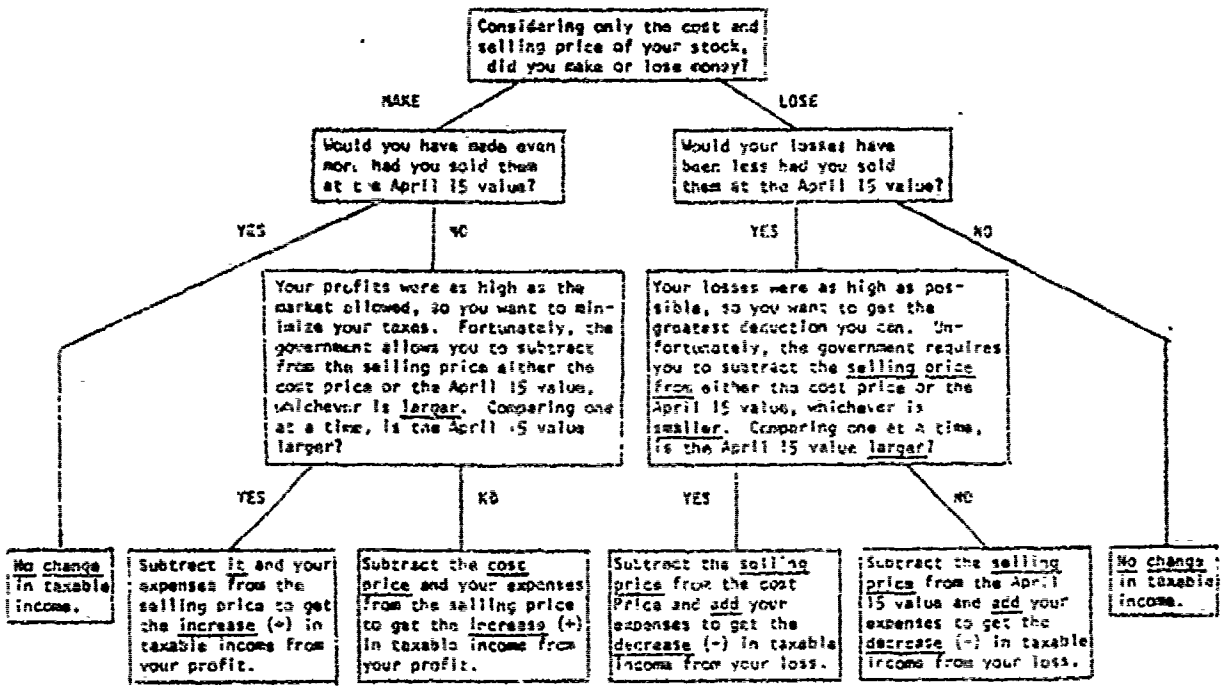


Figure 3.

"Logical" algorithm for solving tax problems (Schmid & Gerlach, 1978).

(individual discriminators and operators), but also the structural cohesiveness of the algorithm as a whole. This cohesiveness leads to a logical formation which aids learning and retention.

Principles drawn from cognitive research do appear to transfer to the more regimented characteristics of algorithmic learning. This finding, while not surprising, adds an important link to the connection between learning studies dealing primarily with word, sentence, and paragraph stimuli and the type of learning typically required on a training level. Furthermore, the concepts presented regarding the development of algorithms via the model should produce results reciprocal to those drawn from propositional- and schema-based studies (Kintsch & Van Dijk, 1978; Thorndike & Yekovich, 1979, in press). This type of empirical correspondence is essential if instructional design is to keep abreast of developments in the psychological fields. History is replete with examples either of blind acceptance and application of basic research, which ended in disaster, or blatant rejection of its results due to "irrelevance."

It is apparent that only a small part of the model presented above has been put to task, and then only in a post hoc fashion. The advantages of such a modeling exercise are nevertheless obvious. The model supplies the instructional designer with a precise terminology for describing both the content and structure of a learning or teaching algorithm. The guidelines concerning the various parameters of a given algorithm presented above can have immediate instructional consequences for improving learning. Once an algorithm has been defined within the context of the model, it is likely that the computer itself can be programmed to "test" for the optimal configuration of the algorithm and, thus, to assist in improving

it. Finally, when content matter, task requirements, or learner characteristics force instruction out of an algorithmic mold, the as yet unexplored concepts of parallel processing (alluded to above) and their corresponding gate-keeping functions may provide a useful model for the study of human memory, with a consequent improvement in instructional design and development.

Reference Notes

1. Mantel, M. M. Modeling user behavior in computer learning tasks. Paper presented at the annual meeting of the American Educational Research Association. San Francisco, April 1979.

2. Personal communication, 1976.

3. Schmid, R. F. Prior knowledge, content familiarity and the comprehension of natural prose. Unpublished dissertation, Arizona State University, 1977.

References

Anderson, J. R., and Bower, G. H. Human associative memory. Washington, D.C.: V. H. Winston and Sons, 1973.

Bung, K. Programmed learning and the language laboratory. London: Longmac Ltd., 1968.

Bung, K. A cybernetic approach to programmed language instruction. Educational Media International, 1971, 4, 1-8.

Bung, K. & Carrasco, M. J. Quasi-Algorithms and the teaching of grammar. Educational Technology, 1977, 17, 48-52.

Feigenbaum, E. A. Simulation of verbal learning behavior. In E. A. Feigenbaum and J. Feldman (Eds.), Computers and Thought. New York: McGRaw-Hill, 1963.

Gerlach, V. S., Reiser, R. A., and Brecke, F. H. Algorithms in learning, teaching, instructional design. Technical Report No. 51201, Bolling AFB, D.C., OSR, 1975.

Kintsch, W., and Van Dijk, T. Toward a model of text comprehension and production. Psychological Review, 1978, 85, 363-394.

Landa, L. N. Algorithms in learning and instruction. Englewood Cliffs, N.J.: Educational Technology Publications, 1974.

Landa, L. N. Instructional regulation and control. Englewood Cliffs, N.J.: Educational Technology Publications, 1976.

Lansky, M. Learning algorithms as a teaching aid. RECALL: Review of educational cybernetics and applied linguistics, 1969, 1, 81-89.

Miller, G. A. The magic number seven plus or minus two: Some limits on our capacity for processing information. Psychological Review, 1956, 63, 81-97.

Newell, A., & Simon, H. A. Computer simulation of human thinking. Science, 1961, 460, 2011-2017.

Paivio, A. Imagery and Verbal Processes. New York: Holt, Rinehart and Winston, Inc., 1971.

Pask, G. Conversation, Cognition and Learning. Amsterdam: Elsevier, 1976.

Quillian, M. R. Semantic memory. In M. Minsky (Ed.), Semantic Information processing. Cambridge: MIT Press, 1968.

Scandura, J. M., Durnin, J., Ehrenpreis, W., and Luger, G. Algorithmic approach to mathematics: concrete behavioral foundations. New York: Harper & Row, 1971.

Schmid, R. F. and Gerlach, V. S. The application of algorithms to instructional systems. U.S.A.F. Office of Scientific Research, Technical Report No. 70831, Arizona State University, August 1977.

Schmid, R. F., & Gerlach, V. S. Principles for developing algorithmic instruction. U.S.A.F. Office of Scientific Research, Technical Report No. 81201, Arizona State University, 1978.

Schmid, R. F., Portnoy, R. C., & Burns, K. Using algorithms to assess comprehension of classroom text materials. Journal of Educational Research, 1976, 69, 309-312.

Snow, R. E. Theory and method for research on aptitude processes. TR #2. Aptitude Research Project, School of Education, Stanford University, 1976.

Thorndike, P. W., & Yekovich, F. R. A critique of schemata as a theory of human story memory. Poetics, 1979, in press.